**6-5.修改旅行售货员问题的分支限界法：**

代码如下：

#include <iostream>

#include <queue>

#include <cfloat>

using namespace std;

struct Node {

float lcost; //子树费用的下界

float rcost; //x[s:n-1]中顶点最小出边费用和

float cost; //当前费用

int s; //根结点到当前结点的路径为x[0:s]

int\* x; //当前路径，x[s+1:n-1]待搜索

Node(float a, float b, float c, int d, int\* e) : lcost(a), rcost(b), cost(c), s(d), x(e) {}

bool operator <(const Node& node)const {

return node.lcost < lcost;

}

};

const int n = 5; //图G的顶点个数

int bestp[n]; //最优解

//邻接矩阵

float a[n + 1][n + 1] = {

0, 0, 0, 0, 0, 0,

0, -1, 5, 61, 34, 12,

0, 57, -1, 43, 20, 7,

0, 39, 42, -1, 8, 21,

0, 6, 50, 42, -1, 8,

0, 41, 26, 10, 35, -1

};

//旅行售货员问题的优先队列式分支限界法

float bbTSP(void) {

//求最小出边费用

float minOut[n + 1];

float minSum = 0;

for (int i = 1; i <= n; i++) {

minOut[i] = FLT\_MAX;

for (int j = 1; j <= n; j++) {

if (a[i][j] > 0 && a[i][j] < minOut[i])

minOut[i] = a[i][j];

}

if (minOut[i] == FLT\_MAX)

return FLT\_MAX; //无回路

minSum += minOut[i];

}

//初始化路径

int\* x = new int[n];

for (int i = 0; i < n; i++)

x[i] = i + 1;

//初始化小顶堆

priority\_queue<Node> MinHeap;

Node node(0.f, minSum, 0.f, 0, x);

float bestc = FLT\_MAX; //最优值

bool exist = true;

//搜索排列树

while (node.lcost < bestc) {

x = node.x;

//当前扩展结点是叶结点的父结点

if (node.s == n - 2) {

if (a[x[n - 2]][x[n - 1]] > 0 && a[x[n - 1]][1] > 0 && node.cost + a[x[n - 2]][x[n - 1]] + a[x[n - 1]][1] < bestc) {

bestc = node.cost + a[x[n - 2]][x[n - 1]] + a[x[n - 1]][1];

for (int j = 0; j < n; j++)

bestp[j] = x[j];

}

}

else {

//产生当前扩展结点的子结点

for (int i = node.s + 1; i < n; i++) {

if (a[x[node.s]][x[i]] > 0) {

float cost = node.cost + a[x[node.s]][x[i]];

float rcost = node.rcost - minOut[x[node.s]];

float lcost = cost + rcost;

//子树可能含最优解，结点插入小顶堆

if (lcost < bestc) {

int\* xx = new int[n];

for (int j = 0; j < n; j++)

xx[j] = x[j];

xx[node.s + 1] = x[i];

xx[i] = x[node.s + 1];

MinHeap.push(Node(lcost, rcost, cost, node.s + 1, xx));

}

}

}

}

delete[] x;

//取下一扩展结点

if (!MinHeap.empty()) {

node = MinHeap.top();

MinHeap.pop();

}

else {

exist = false;

break;

}

}

//释放内存

if (exist)

delete[] node.x;

while (!MinHeap.empty()) {

node = MinHeap.top();

MinHeap.pop();

delete[] node.x;

}

return bestc;

}

//测试程序

int main(void) {

cout << "最小费用：" << bbTSP() << endl;

cout << "路径：";

for (int i = 0; i < n; i++)

cout << bestp[i] << "->";

cout << bestp[0] << endl;

return 0;

}